VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE OBJEKTŮ POMOCÍ KINECTU

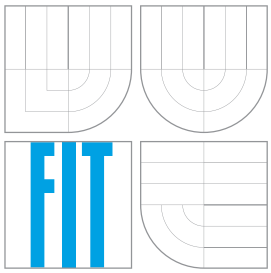
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ NĚMEC

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE OBJEKTŮ POMOCÍ KINECTU

OBJECT DETECTION USING KINECT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ NĚMEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL ŠPANĚL, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Tato práce se zabývá problémem detekce objektů pomocí senzoru Microsoft Kinect v oborů počítačového vidění. Cílem bylo zhodnotit současné metody detekce objektů využívající hloubkovou mapu (RGB–D senzor). Práce se zabývá prostředím mračna bodů a metodou Viewpoint Feature Historgam. Také popisuje využití binárních klasifikátorů v rámci rozpoznání objektů. Návrh detekce objektů byl v práci realizován a byly na ní prováděné experimenty.

Abstract

This paper addresses the problem of object recognition using Microsoft Kinect in the field of computer vision. The objective of this work was to evaluate current methods of detection of objects using depth map (RGB–D sensor). The work deals with the enviroment of point cloud and Viewpoint Feature method. It also describes the use of binary classifier in the context of object recognition. Object detection was implemented and performed experiments with it.

Klíčová slova

Detekce objektů, Microsoft Kinect, mračno bodů, Point Feature Histogram, PFH, Fast Point Feature Histogram, FPFH, Viewpoint Feature Histogram, VFH, Support Vector Machienes, SVM

Keywords

Object detection, Microsoft Kinect, point cloud, Point Feature Histogram, PFH, Fast Point Feature Histogram, FPFH, Viewpoint Feature Histogram, VFH, Support Vector Machienes, SVM

Citace

Lukáš Němec: Detekce objektů pomocí Kinectu, bakalářská práce, Brno, FIT VUT v Brně, 2014

Detekce objektů pomocí Kinectu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Španěla, Ph.D.

.....
Lukáš Němec
20. května 2014

Poděkování

Tímto bych chtěl poděkovat panu Ing. Michalovi Španělovi, Ph.D. za vedení při práci na této bakalářské práci. Za jeho rady při pravidelných konzultacích, které vedly ke svědomitému přístupu a dokončení této práce včas.
Děkuji

Obsah

Kapitola 1

Úvod

Počítačové vidění je jak vědou, tak i technologií usilující o vytvoření stroje, který si osvojuje vlastnosti vidět a vnímat okolí. Tyto stroje pak mohou být důležitým zdrojem informací v moderní informační době. Jako jsou informace pro modelování objektů při obrazové kontrole kvality objektů nebo i při lékařské analýze obrazu. Využití těchto strojů ale nekončí u možnosti kontrol nebo při využití různých řídicích systémů, ale umožňují také interakci mezi člověkem a strojem. Příkladem stroje umožňující tuto interakci je senzor Kinect od firmy Microsoft využívaný pro ovládání počítačových her.

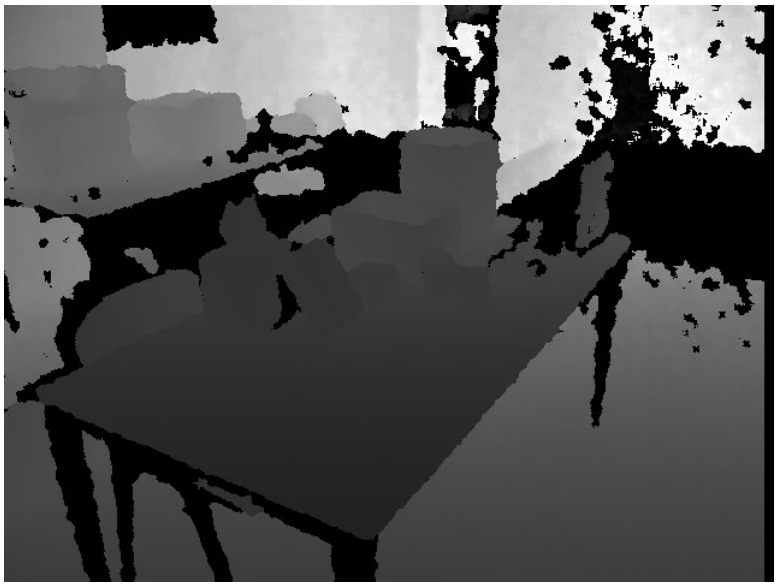
Tato bakalářská práce popisuje způsob detekci objektů s využitím informace hloubkové mapy, která je získávána ze senzoru Kinect. Cílem této práce je navrhnout a implementovat takovýto detektor. Jako možnost jiné reprezentace hloubkové mapy bylo zvoleno mračno bodů, ve kterém se vyhledávají objekty. Detekce objektů byla navrhnuta pomocí metody Euclidean Cluster Extraction a odstraněním ploch. Samostatné objekty jsou popisovány deskriptorem vypočítaným metodou Viewpoint Feature Histogram. Jenž je základem pro klasifikaci objektů pomocí binárního klasifikátoru Support Vector Machines.

Práce je členěna do několika kapitol, jež jsou organizovány následovně. Kapitola 2 pojednává o možnostech způsobů popisu objektů, který je snímán hloubkoměrem Kinect. Také vysvětluje způsob klasifikace objektů, čímž dává teoretický základ pro pochopení dalších částí technické zprávy. V kapitole **??** jsou popisovány existující nástroje, které umožňují využití již implementovaných algoritmů pro vytvoření deskriptorů objektů a jejich klasifikaci. Následující kapitola 4 se již zabývá návrhem detektoru, jeho jednotlivými částmi a ukazuje jejich návaznost na sebe. Kapitolu 5 tvoří popis implementace a ukazuje možnosti využití již existujících nástrojů pro tvorbu detektoru. V předposlední kapitole 6 jsou ukázané vlastnosti klasifikátorů a jejich schopnost rozpoznat jednotlivé objekty a jejich síla oproti ostatním. V poslední kapitole 7 jsou popsány dosažené výsledky, možnosti dalšího vývoje a zhodnocení celé práce.

Kapitola 2

Popis a rozpoznávání objektů

Tato kapitola přibližuje problematiku detekce a rozpoznávání objektů z dat získaných například senzorem Microsoft Kinect, který nám poskytuje hloubkovou mapu snímaného okolí. Popisuje převod získané hloubkové mapy do mračna bodů a ukazuje různé způsoby a problémy popisu povrchu objektů v mračnu bodů.



Obrázek 2.1: Hloubková mapa z Kinectu.

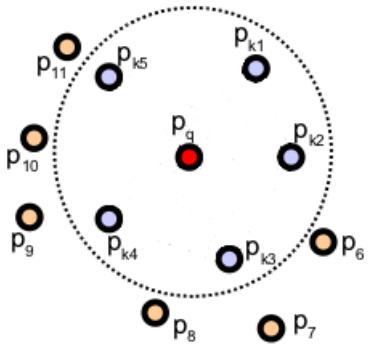
2.1 Mračno bodů

Mračno bodů je vlastně uspořádaná nebo neuspořádaná množina bodů. Jednotlivé body jsou zde reprezentovány jako datové struktury se svými vlastnostmi. Tyto vlastnosti mohou být různé, ať již od hodnot RGB či jejich kategorie. Nejdůležitější vlastnosti jsou určující souřadnice x, y, z, kde z souřadnice se získá pomocí hloubkové mapy. Pomocí těchto souřadnic je možno tyto body zařadit do mračna a vytvořit tak 3D reprezentaci snímaného okolí.

Sousední body

V kapitole 2.1 je popsáno, že mračno bodů může být neuspořádaná množina, a většinou je tak i reprezentováno. Zde nastává problém určení nejbližšího bodu či množiny okolních bodů. Jak se dozvíme v dalších kapitolách tato informace je důležitá především pro výpočet normál pro popsání povrchu.

Určení sousední množiny bodů, daného bodem p_q, je relativní. Velice zde závisí na zadané hodnotě d, nebo-li poloměru abstraktní koule v 3D prostoru ohraničující prostor, ve kterém se nachází sousední body bodu p_q. Všechny body nacházející se v tomto sousedství tedy musí splňovat, že jejich prostorová vzdálenost od bodu p_q je menší než hodnota poloměru nebo rovna d.



Obrázek 2.2: 2D reprezentace sousedních bodů bodu p_q (uvnitř kružnice) a bodů, které nejsou sousední (vně kružnice). Převzato z [2].

2.2 Point Feature Histogram desriptor

Mračno bodů je tedy reprezentováno množinou bodů, které mají svoje vlastnosti. Mezi jednotlivými body tak není nic a není možno jen z údajů vlastností těchto bodů udělat přesnou rekonstrukci povrchu mračna.

Jelikož je pro detekci objektů velice důležité určit tvar povrchu, aby bylo možné objekt rozpoznat. A následně přiřadit nějaké třídě objektů, musí se tento problém adresovat. Je tedy potřeba vytvořit, za pomoci bodů, reprezentaci povrchu objektů. Jednou z možností je vytvořit deskriptor povrchu objektů pomocí metody Point Feature Histogram (PFH) [8].

Popis povrchu objektu v mračnu bodů

Pro popis povrchů různých objektů je možno použít mnoha různých metod. Jednou z lehčích metod je použití normál bodů na povrchu objektu. Tato metoda je často využívána v grafických aplikacích, například při užití světelného zdroje, který generuje stíny či jiné vizuální efekty.

Určení normál

Princip této metody je stanovení normály k bodu v mračnu na základě aproximace problému odhadu normály z rovinné tečny k ploše. Plocha má výsledný tvar nejmenšího čtverce

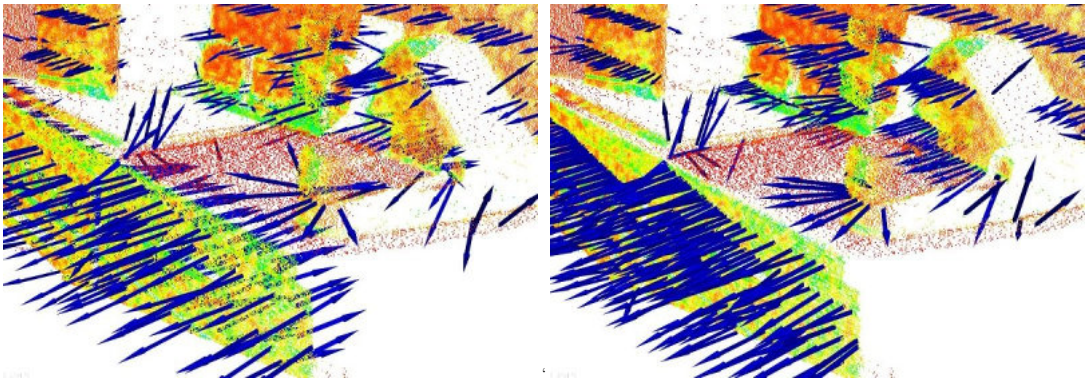
vhodné plochy pro daný odhad. Plocha je zde daná pomocí bodu p a vektoru $\vec{n}$, kde hodnota p a $\vec{n}$ je spočítaná pomocí nejmenšího čtverce, tak že vzdálenost sousedícího bodu od plochy je nula[8]. Řešení je tak redukováno o analýzu vektorů a hodnot kovarianční matice, vytvořené ze sousedních bodů bodu p, definovanou jako:

$$C = 1/k \sum_{i=1}^{k_i} \zeta_i * (p_i - \overline{p}) * (p_i - \overline{p})^T, C * v_j = \lambda_j * v_j, j \in \{0, 1, 2\} \tag{2.1}$$

Kde ζ reprezentuje možnou váhu bodu (většinou hodnotu 1), k je počet sousedů v sousedství p_i daného bodu, p je těžiště nejbližších sousedů, λ_j je j-té číslo matice a v_j je j-tý vektor [9].

Směr normál

U takto vypočítaných normál není možné určit orientaci dané normály [9]. V obrázku 2.3 je znázorněno, že některé normály jsou v obráceném směru, než v jakém je potřeba pro přesný odhad povrchu. Tyto normály směřují opačným směrem. Proto je potřeba tyto normály najít a otočit je vůči senzoru, tedy zaměnit znaménko za opačné. Základní myšlenka opravy tohoto problému je, že všechny normály testujeme tak, aby součin hodnoty vektoru a rozdílu mezi hlediskem a bodem, ze kterého normála vychází, musí být menší než nula. Normály, které tento vztah splňují, směřují stejným směrem jako směr senzoru, což je špatně. Naopak normály, které tento vztah nesplňují a jejich součin je větší než nula směřují směrem proti hledisku, jenž je správný směr.



Obrázek 2.3: Obrázek vlevo ukazuje normály, u kterých nejsou poopraveny znaménka směru a obrázek vpravo ukazuje normály, které již mají upravené směry. Převzato z [2].

Velikost sousedství

Jak je definováno v rovnici 2.1 je potřeba mít pro výpočet normál, přesněji hodnot kovariační matice, sousední body daného bodu. A jak bylo řečeno v kapitole 2.1 velikost tohoto sousedství je dána poloměrem abstraktní koule, v které se nachází sousední body. A tato hodnota může velice ovlivňovat výslednou hodnotu deskriptoru povrchu. Velké hodnota poloměru sousedství má za následek počítání normál z více okolních hodnot, tím pak výsledná normála je ovlivněna i hodnotami bodů, které pak nejsou zrovna pro výpočet potřeba. Následkem takového výpočtu může být rozmazání hran a potlačení detailů, které můžou mít

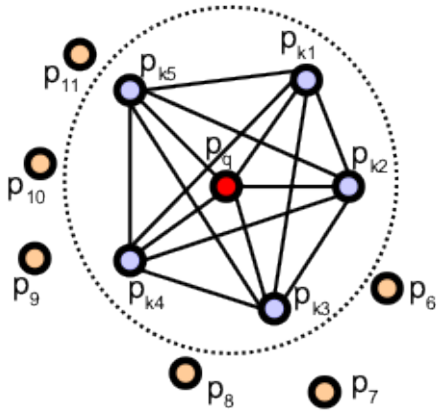
za následek špatné rozeznání objektu. Naopak, pokud vezmeme poměrně dobrou velikost sousedství, nebude vznikat tolik odchylek od nepotřebných bodů a budou viditelné i drobné hrany po celém mračnu. Ve výsledku tedy záleží na aplikaci, v kterém rozlišení pracuje.

Point Feature Histogram

Pro výpočet výsledného popisu využívá metoda PFH povrchové normály a odhad zakřivení základních geometrických tvarů v okolí právě počítaného bodu [9]. Přestože metoda PFH je docela rychlá, i když nedosahuje rychlosti počítání v reálném čase nebo v skoro reálném čase. Jde říct, že je metoda jednoduchá a není schopna zachytit příliš mnoho detailů. To je způsobeno tím, že se počítá s velmi málo hodnotami, které jsou většinou pouze odhadem z okolních hodnot.

Vztahy mezi body

Výsledkem PFH je formulace multi-dimenzionálního histogramu hodnot, které reprezentují geometrické vlastnosti a odhad normál v sousedství daného bodu mezi všemi dvojicemi bodů nacházející se sousedství daného bodu (obrázek 2.4 představuje možnou podobu vztahu v sousedství bodu p_q). Jednoduše řečeno snaží se zachytit co nejlépe povrchové změny ze vzorku, při zohlednění veškeré interakce mezi směry odhadovaných normál povrchu. Výsledek je tedy velice závislý na kvalitě odhadovaných normál.

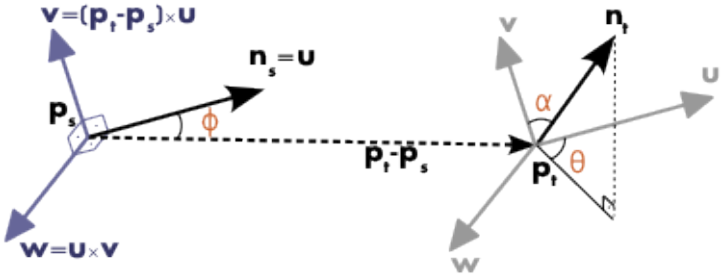


Obrázek 2.4: Zobrazení všech vztahů dvojic v sousedství bodu p_q. Převzato z [2].

Výpočet Point Feature Histogramu

Výpočet relativních rozdílu mezi dvěma body p_s a p_t a jejich normálami n_s a n_t lze definovat jako fixní souřadnice vektorů $\vec{u}$, $\vec{v}$ a $\vec{w}$ pro daný bod. To umožňuje vyjádřit rozdíl normál jako trojici úhlu pro každý bod. Jak je znázorněno v obrázku 2.5 tuto trojici tvoří úhly α, ϕ a θ. Jak je vidět z obrázku hodnota úhlu je α je závislá na vztahu mezi vektorem $\vec{v}$ a hodnotou normály n_t. Hodnota úhlu ϕ je zase určena vztahem mezi vektorem $\vec{u}$ a podílem hodnoty vektorů $p_t - p_s$ a hodnotou d, což je hodnota vzdálenosti mezi dvěma určenými body. Poslední úhel θ je dán hodnotou arctan výsledné hodnoty součinu vektoru $\vec{w}$ a normály n_t a součinu vektoru $\vec{u}$ a normály n_t. K této trojici úhlu se ještě počítá hodnota jejich vzdálenosti d, která je vzdálenosti bodu od kamery a nemá moc velký význam pro

tvorbu histogramu. Přesto tato čtveřice $(\alpha, \phi, \theta, d)$ se počítá pro každou dvojici bodů ze sousedství daného bodu. Tato skutečnost redukuje obecných 12 hodnot (souřadnice x, y, z a vektory $\vec{u}$, $\vec{v}$, $\vec{w}$) na pouhé 4 hodnoty [9][8].



Obrázek 2.5: Zobrazení závisloti dvou bodu při výpočtu PFH. Převzato z [2].

Tyto 4 hodnoty se využívají k sestavení výsledné reprezentace PFH daného bodu. Všechny tyto hodnoty všech bodů jsou pak použity při tvorbě histogramu. Jednotlivé hodnoty z dané čtveřice jsou rozděleny na x podintervalů a počítá se hodnota výskytů bodů v těchto intervalech. Jelikož jsou 3 hodnoty z této čtveřice úhly a jsou potřebné pro popis povrchu, je možno tyto hodnoty jednoduše normalizovat na stejné intervaly na jednotkové kružnici pomocí geometrických funkcí. Tyto normalizované hodnoty jsou pak počítány do intervalů tvořící výsledný histogram [9][8].

Odchylka výpočtu

Jelikož výsledná signatura je hodně závislá na odhadu normál, může nastat, že se do momentálního výpočtu dostane nechtěná odchylka od minulého výpočtu. To vede k malým odlišnostem ve výsledcích. Tento problém by mohl mít obzvlášť důsledek, pokud by se hodnoty nacházeli na okrajích rozdělovacího prahu dvou intervalů histogramu. Hodnoty by pak mohli být podle algoritmu započítávány do různých intervalů. Řešením tohoto problému je celkem jednoduché a spočívá v tom, aby výsledný histogram pro různé objekty nebyl rozdělen na sudý počet intervalů. Jednoduše řečeno nechceme, aby při rozdělování hodnot do intervalů bylo u hranatých objektů pouze o 90 °, tedy aby se histogram například krabice neskládal pouze z hodnot pod a nad 90 °. Rozdělením na lichý počet se tedy dosáhne, že tyto objekty je možno mnohem lépe popsat a tím také bude histogram více odolný vůči dalším možným odchylkám a nebude se ztrácet ani při popisu jiných primitivních geometrických tvarů.

Stručný algoritmus pro popis povrchu pomocí Point Feature Histogram

- pro každý bod p, urči všechny sousedy p v jeho sousedství určené hodnotou poloměru d,
- pokud bod p nemá povrchovou normálu, aproximuje se normála pomocí nejmenšího čtverce vhodného pro dané sousedství bodu p,
- pro všechny normály, které byly získány, se zkontroluje směr dané normály vůči hledisku a upraví jejich orientaci,

- pro každý bod p_i v sousedství p a jejich normály se spočítají rozdíly jejich úhlů,
- následně se počítá výskyt jejich hodnot na určených intervalech, z nichž se složí výsledný histogram popisující povrch objektu.

2.3 Fast Point Feature Histogram descriptor

Pokud je potřeba dosáhnout popisu objektu v reálném čase nebo aspoň v rozumném časovém intervalu blížícímu se k reálnému času, je PFH velice nevhodné se svou složitostí $O(nk^2)$, kde n je počet bodů v mračnu a k je počet sousedů daného bodu. Výpočetní doba například pro normální hrníček by v tomto případě byla příliš velká, neboť normální hrníček získán z dat Kinectu a je reprezentován asi 2000 body a pro každý se musí počítat výsledná hodnota do deskriptoru. Je tedy zřejmé, že pro detekci v reálném čase není PFH zcela vhodná. Jedno z řešení je použít různé filtry pro snížení počtu bodů v mračnu, ale pouze za cenu méně přesnějšího vypočítaného histogramu. Nebo existuje další možnost a tou je metoda Fast Point Feature Histogram (FPFH) [7], jenž je zjednodušenou verzí PFH. Tato metoda redukuje výpočetní dobu PFH z $O(nk^2)$ na $O(nk)$ a přesto jí zůstává většina důležitých aspektů PFH metody [7].

Výpočet Fast Point Feature Histogram

FPFH se snaží urychlit výpočet výsledného PFH deskriptoru a zároveň s tím si ponechat sílu a vlastnosti PFH. Proto je algoritmus FPFH lehce odlišný od PFH a provádí se ve dvou krocích:

- pro každý bod p v mračnu bodů a jeho sousedů je vypočítána n-tice $< \alpha, \theta, \phi >$ jak je popsáno v kapitole o PFH. Toto se nazývá Simplified Point Feature Histogram (SPFH) [9],
- v druhém kroku, bude pro každý bod jeho sousedství použito SPFH hodnot pro rozhodnutí finální hodnoty pro histogram bodu p podle:

$$FPFH(p) = SPFH(p) + 1/l \sum_{i} = 1^k 1/w * SPFH(p_k) \tag{2.2}$$

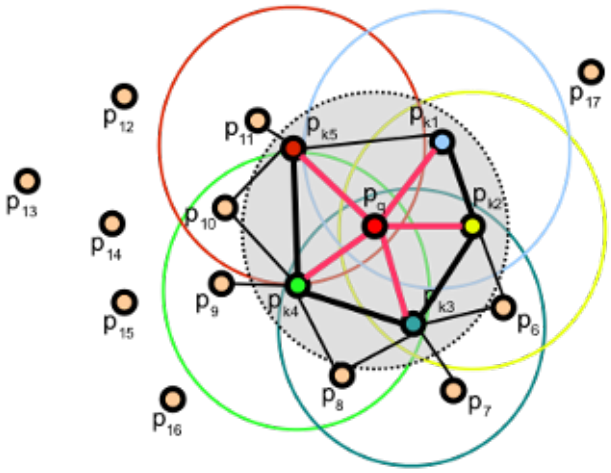
kde w je reprezentace vzdálenosti mezi p a jeho sousedním bodem p_k v daných jednotkách. Tedy pro každý bod p z mračna bodů, algoritmus nejdříve vypočítá hodnoty SPFH vytvořením páru mezi daným bodem a jeho sousedy. Tohle je opakováno pro všechny body v daném mračnu. Pak je určována jejich hodnota na základě SPFH hodnot pro daný bod p použitím hodnot SPFH jeho sousedního bodu p_k, tím vytvoření FPFH hodnoty pro bod p [7].

Výsledné hodnoty jsou tedy závislé na hodnotách okolních bodů a ty jsou zase závislé na hodnotách svého sousedství. Obrázek 2.6 zobrazuje vztah bodu p s okolními body a jejich sousedstvím. Okolní body, které jsou přímo spojené s bodem p, jsou spojeny také se svým sousedstvím. Výsledný histogram je vyhodnocen společně s histogrami okolních bodů.

Rozdíl mezi FPFH a PFH

Aby FPFH bylo schopno dosáhnout vyhodnocení popisu povrchu v reálném čase, musí se od předešlé metody PFH lišit. Základní rozdíly FPFH proti PFH jsou:

- FPFH nepropojí všechny sousedy a tím pádem mu chybí některé hodnoty párů, které by mohli být důležitými a přispět k zachycení povrchu okolo daného bodu,
- PFH modely jsou přesným zachycením a určením povrchu v okolí bodu, zatímco FPFH získává i hodnoty mino poloměr daného sousedství,
- kvůli přehodnocování hodnot FPFH kombinuje hodnoty SPFH a přehodnotí některé hodnoty sousedících párů,
- celková složitost, tím pádem rychlost výpočtu FPFH je vysoce redukována a je možno ji použít v aplikacích pracujících v reálném čase,
- výsledný histogram je zjednodušen nesouvžtažností hodnot, to je jednoduše vytvořeno nezávislými histogramy vlastností, jeden pro každou dimenzi a korelací dohromady.



Obrázek 2.6: Zobrazení všech vztahů v sousedství bodu p_q v metodě FPFH. Převzato z [2].

PFH optimalizováním může být ještě vylepšeno, pokud se soustředíme na souvztažnost vlastností v histogramu. Doposud se počítalo s histogramy kategorií, které byly dány vztahem b^d, kde b je počet podkategorních intervalů ve vlastnostech hodnot vzdálenosti a d je počet vybraných vlastností. Podle [9] to může vést k tomu, že výsledný histogram bude obsahovat velice mnoho hodnot nula, což vede k redukování hodnot informací v histogramu a některé podkategorie nebudou mít nikdy hodnotu. Zjednodušit tedy tuto nesouvztažnost hodnot je jednoduché a je potřeba vytvořit jednotlivé vzdálenostní histogramy. Jeden pro každou dimenzi a spojit je dohromady.

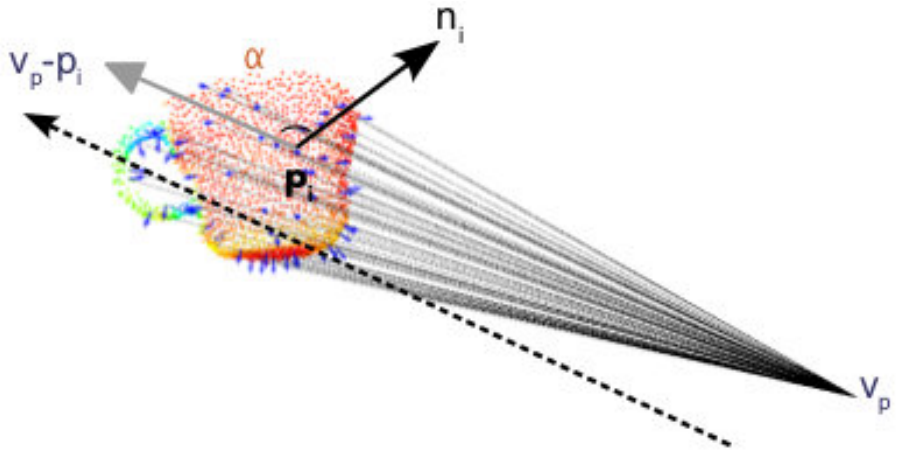
2.4 Viewpoint Feature Histogram descriptor

Metoda podobně jak PFH nebo FPFH pro popisování povrchu objektu a je také rozšířením FPFH, neboť jej pro své výpočty využívá společně se statistikami relativních úhlů normál. Výhodou VFH je především rychlost výpočtu, která je téměř v reálném čase. A na rozdíl od FPFH, jehož algoritmus má složitost $O(nk)$, kde n je počet bodů v mračnu a k je počet sousedů daného bodu. Má VFH složitost pouze $O(n)$. Má tak rychlejší výpočetní rychlost, a

přesto stále má velkou sílu v rozpoznávání jakou má FPFH. Společně s hlediskem rozptylu má také odolnosti vůči měřítku objektů.

Myšlenka

Hlavní myšlenka VFH je zkombinovat směr pohledu kamery senzoru (V_p) a relativní úhly normál spočítané pomocí FPFH, jak je znázorněno na obrázku 2.7. Tedy rozšířit FPFH, tak aby udělal odhad pro celé mračno a spočítal další statistiky mezi V_p a odhadem normál každého bodu [6]. V_p komponenta je počítána sbíráním histogramů těch úhlů, jenž jsou vytvořeny vektorem směru V_p a normály bodu. Přesněji řečeno se jedná o úhly mezi směrem centrálního V_p do všech normál. Nejedná se tedy o viditelné úhly normál, neboť by pak výsledné hodnoty nebyly odolné vůči různým měřítkům objektů. Objekt by pak musel být neustále ve stejné vzdálenosti od hlediska či by se muselo mít několik různých rozlišení stejných objektů pro jeho detekci. Druhá komponenta měří relativní hodnoty úhlů stejně jako FPFH metoda (více v kapitole 2.2), ale s rozdílem, že je počítá vůči V_p.

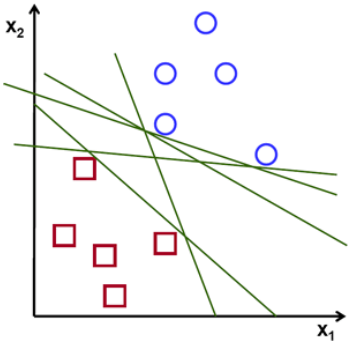


Obrázek 2.7: Zobrazení směru pohledu kamery a komponenty pro výpočet úhlu v VFH. Převzato z [2].

2.5 Support Vector Machine

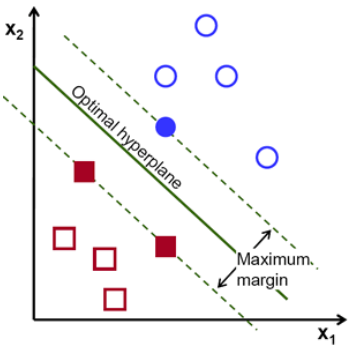
Support vector machines (SVM) je metoda strojového učení. Využívá se ke klasifikaci, která v příznacích rozhoduje na základě předem natrénovaných dat. Snaží se najít nadrovinu, tak aby mohl rozdělit body na jednu či druhou stranu. Na popis takové nadroviny stačí pouze nejbližší body, kterých je obvykle málo, tyto body se nazývají podpůrné vektory (angl. support vectors). Snaží se tedy najít optimální nadrovinu, tu je možné definovat jako nadrovinu která rozděluje prostor tak, že body leží na opačných poloprostorech a hodnota vzdálenosti nejbližších bodů od roviny je co největší [1]. Tato metoda je přirozeně binární, rozděluje je jedinou nadrovinou, tedy rozděluje je na dvě různé třídy.

V obrázku 2.8 jsou znázorněna různá řešení rozdělení pomocí lineární funkce, tedy nadroviny. Je tedy potřeba určit tu optimální nadrovinu, která rozdělí tento prostor na dva poloprostory. Poloprostory pak budou obsahovat jednotlivé body. Jak je psáno v předešlém



Obrázek 2.8: Ukázka více možných řešení problému. Převzato z [5].

odstavci je potřeba vyhledat optimální nadrovinu. Ta by měla rozdělit body, tak aby minimální hodnota vzdálenosti nejbližších bodů od určené nadroviny byla co největší, jak je zobrazeno v obrázku 2.9.



Obrázek 2.9: Optimální nadrovina. Převzato z [5].

Více třídní Support Vector Machines

Jelikož je SVM ve své podstatě binární, rozděluje na jeden ze dvou poloprostorů. Vystává problém s klasifikací na více tříd. Ten se řeší pomocí rozdělení více tříd na několik problémů binárních. Jedním z takových přístupů může být rozdělení algoritmu na jeden proti všem. Kde klasifikátor s největším výstupem je prohlášen za vítěze a bere vše. Nebo je možno použít metodu jeden proti jednomu, kde je rozhodnuto podle největšího počtu vítězství v jednotlivých duelech mezi jednotlivými klasifikátory.

ROC křivka

Je nástroj pro hodnocení, optimalizaci a grafické znázornění chování binárního klasifikačního systému (testu), který ukazuje vztah mezi specificitou a senzitivitou daného testu nebo detektoru pro všechny přípustné hodnoty prahu [11].

Při binární klasifikaci určuje klasifikační pravidlo, zda jedinec či objekt patří do třídy pozitivních nebo negativních objektů. Můžeme se setkat se dvěma klasifikačními pravidly. První pravidlo zařadí objekt jako pozitivní, pokud výsledná hodnota je větší než hodnota

prahu. Druhé pravidlo přiřadí objekt zase k negativních, pokud je hodnota menší nebo rovna prahu rozhodování.

Pokud tedy je známa klasifikační třída objektu a jeho skutečná třída je možno objekt zařadit do jedné ze 4 kategorií:

- True Positive (TP) - jedinci, kteří jsou ve skutečnosti pozitivní a klasifikační pravidlo je zařadilo mezi pozitivní,
- False Negative (FN) - jedinci, kteří jsou ve skutečnosti pozitivní avšak klasifikační pravidlo je zařadilo mezi negativní,
- False Pozitive (FP) - jedinci, kteří jsou ve skutečnosti negativní avšak klasifikační pravidlo je zařadilo mezi pozitivní,
- True Negative (TN) - jedinci, kteří jsou ve skutečnosti negativní a klasifikační pravidlo je zařadilo mezi negativní.

Pro jednotlivé kategorie se počítá počet přiřazení objektů ke kategorii. Tato četnost přiřazení je závislá na dělicím prahu. Jestliže se dělící prah, změní, musí se přepočítat hodnoty v jednotlivých kategoriích.

Z definice ROC křivky se ale jedná o vztah mezi specificitou a senzitivitou, tedy poměrem správně pozitivních porovnání ku všem pozitivním případům a poměrem správně negativních porovnání ku všem negativním případům. Hodnoty specificity a senzitivity se tedy vypočítají takto:

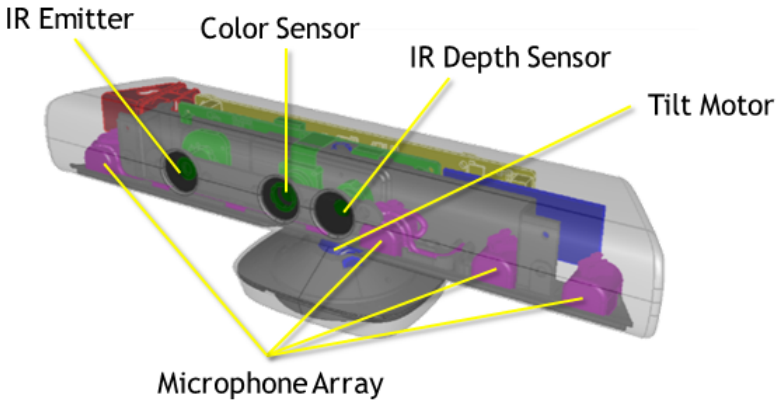
- senzitivita: $\frac{TP}{TP+FN}$, neboli míra Skutečně Pozitivních (True Positive Rate),
- specificita: $\frac{TN}{FP+TN}$, neboli míra Skutečně Negativních (True Negative Rate).

Pro vytvoření dvourozměrného grafu ROC křivky je potřeba na osu x nanést pravděpodobnost špatného zařazení skutečně negativních objektů $(1 - specifita)$. Na osu y zase se nanese hodnota pravděpodobnosti správného zařazení pozitivních objektů $(senzitivita)$. Tyto hodnoty se počítají pro všechny hodnoty dělícího prahu a pro každou hodnotu dostaneme právě jeden bod.

Kapitola 3

Microsoft Kinect

Zařízení Kinect je senzor, který pomocí hloubkové informace snímaného okolí umožňuje interakci se strojem, ke kterému je připojen. Kinect byl vytvořen firmou Microsoft s hlavním zaměřením na interakci uživatele a jejich herní konzole Xbox 360, dá se použít i s operačním systémem Windows. Umožňuje tedy uživateli ovládat Xbox 360 nebo stolní počítač bez nutnosti využití jiných ovládacích prvků pouze pomocí různých mluvených příkazů nebo gesty uživatele, například mávání ruky, skoky apod. Pro Kinect jsou vytvořeny vývojové prostředky, které umožňují vytvářet aplikace v jazycích C++, C# nebo Visual Basic .NET.



Obrázek 3.1: Microsoft Kinect a jeho senzory. Přebráno z [4].

3.1 Základní technické parametry pohybového senzoru Kinect pro konzoli Xbox 360

Kryt Kinectu je trvořen plastem a samotný senzor je schopen vertikálního natáčení pomocí nastavitelného podstavce. Technické parametry Kinectu:

- rozměry bez podstavce – 28 x 6 x 4 cm (šířka, délka, výška),
- hmotnost – 1,57 kg,
- pozorovací úhly Kinectu – 43° vertikálně a 57° horizontálně

Kinect pro umožnění interakci s uživatelem a zachycení jeho příkazů využívá senzory:

- RGB kamera – slouží k zachycování 2D barevných obrázků nebo obličejů uživatelů v rozlišení 1280x960,
- hloubkový senzor – pro zachycení gest uživatelé, tvoří ho emitor infračervených paprsků a senzor pro zachytávání těchto paprsků,
- 4 mikrofony – pro zachycení mluvených příkazů, a také k určení místa zdroje a směr zvukové vlny.

Rozsah pro zachycení uživatele pomocí těchto senzorů je 1,2m – 3,5m a je schopen zachytit 2 aktivní uživatele [4]. Jednotlivé senzory jsou zabudované horizontálně vedle sebe, jak je ukázano v obrázku 3.1.

S vydáním nové generace konzole Xbox One přišel Microsoft i s novým vylepšeným senzorem Kinect pro tuto konzoli. Nová verze tohoto senzoru využívá širokoúhlou kameru s rozlišením 1080p a zpracovává 2GB dat za sekundu při analýze okolí. Je také schopen registrovat až 6 aktivních uživatelů najednou. Nový Kinect dále umožňuje skenovat QR kódy nebo také tepovou frekvenci.

Kapitola 4

Návrh detektoru

Cílem detektoru je detekovat a rozpoznat geometrické tvary objektů pomocí senzoru Microsoft Kinect. Jelikož je možno detekovat cokoliv rozhodl jsem se, že se zaměřím na jednu část možných objektů a nesnažil se naučit detektor rozpoznávat objekty nacházející se různě v okolí. Oblast, na kterou jsem se zaměřil, jsou geometrické tvary objektů, které je možno najít na stole. Mezi ně patří tvary objekty, jako jsou hrnek, láhev, talíř, ale i různá ovoce jako je jablko či banán nebo různá elektronická zařízení jako je kalkulačka či mobilní telefon.

Snímaná scéna by tedy měla obsahovat objekty, které se můžou nacházet na stole. Nejlépe umístěny na stole, který nemá prosklenou plochu. Zde by bylo možné dojít k problému se senzorem, který snímanou plochu nezaznamená a převedená hloubková mapa by neobsahovala plochu stolu. I když ve výsledné implementaci by toto nemusel být až takový problém je zde ale možnost ovlivnění některých klastrů, které by tak mohli být rozšířeny o body, které k nim nepatří.

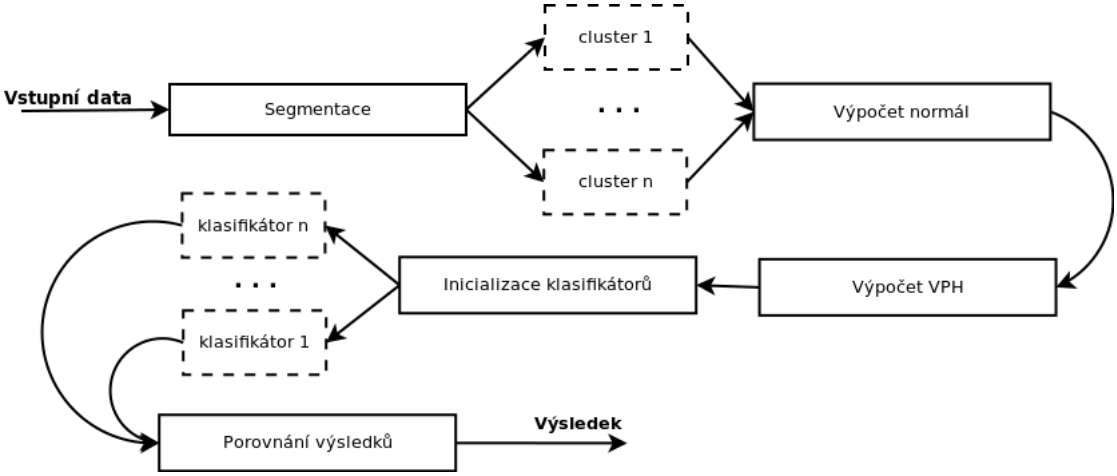
Výsledkem detektoru by tedy měla být informace pro uživatele, která ho bude informovat o detekovaných objektech. Tyto objekty by také měli být pomocí klasifikátorů rozpoznávány a určeny, ke které třídě objektů patří.

4.1 Souhrn výsledného detektoru

Výsledný detektor by se měl skládat z několika různých částí. Každá z těchto částí by měla provádět určitý úkon a její výstup by měl být vstupem pro další část detektoru. Jednotlivými části detektoru jsou:

- Vstupní data z Kinectu nebo ze souboru
- Segmentace na jednotlivé objekty metodou Euclidean Cluster Extraction a odstraněním ploch.
- Výpočet deskriptorů objektů pomocí PFH, FPFH nebo VFH
- Klasifikace objektů do tříd pomocí SVM

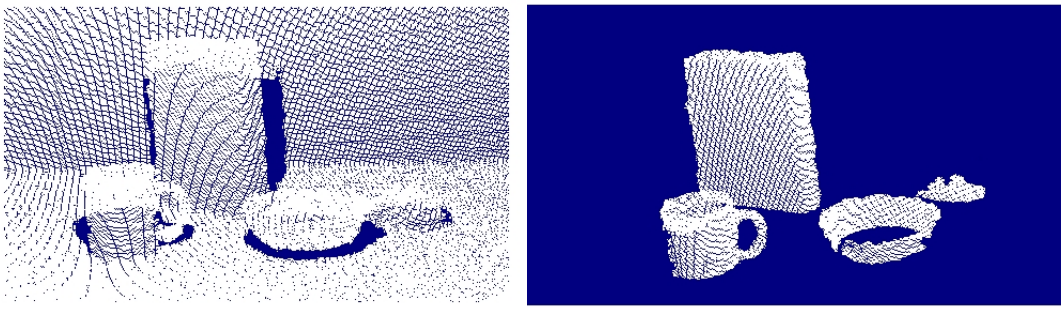
V obrázku 4.1 lze vidět závislosti jednotlivých částí na předchozích výpočtech v detektoru. Také ukazuje cestu, kterou musí vstupní data provést, než se přetransformují na odpovídající výsledek.



Obrázek 4.1: Návrh výsledného detektoru.

4.2 Segmentace mračna

Jelikož je cílenou oblastí detekce objektů na stole ve snímané scéně, je možno si několik věcí ulehčit a vyhledat tak různé části, které v mračnu bodů nejsou potřeba.



Obrázek 4.2: Celé mračno (vlevo) a po odstranění ploch (vpravo) zůstavají jen petenciální objekty.

Velké plochy

Základním objektem, nebo části mračna, kterou si takhle můžeme odstranit, jsou plochy. Přesněji řečeno velké plochy, odstranění malých ploch by mohlo mít za následek odstranění možných cílů detekce. Mezi velké plochy se míní najít plochy, které jsou větší než průměrná velikost ploch ve scéně. Jedná se tedy hlavně o odstranění plochy stolu, na které objekty stojí. Odstranění ploch by mělo mít za následek ponechání pouze možných objektů v mračnu, jak ukazuje obrázek 4.2.

Jednotlivé plochy k odstranění z mračna bodů jsou detekovány pomocí RANSAC (Random Sample Consensus), která odhaduje parametry matematických modelů. Tato metoda produkuje výsledek s určitou pravděpodobností, kterou je možno zvětšit s větším počtem iterací. Algoritmus RANSAC metody se skládá ze dvou kroků:

- nejdříve jsou náhodně vybrány body ze vstupní datové množiny a parametry modelu jsou počítány pouze z množiny náhodně vybraných bodů,
- v druhém kroku metody se kontroluje, které body ze vstupní množiny skládají daný model s parametry odhadnutých v prvním kroku [3].

Skupinu zbylých klastů v nejlepším případě tvoří pouze reálné existující objekty. Často se ale stává, že v teto množině zůstanou zbytky objektů, které budou také tvořit skupinu objektů pro klasifikaci. Může se tak stát, že snímaná scéna je snímána pod úhlem, který zachycuje i nohy stolu či boční hranu. To také může vést k rozšíření skupiny objektů pro klasifikaci. Pak takové objekty, které nejsou cílem detekce a nevytváří ani kompletní objekt zpomalují a zhoršují výsledek následné klasifikace.

Šum v mračnu bodů

Tímto šumem jsou myšleny body, které se vygenerují ze snímané hloubkové mapy, v místech v mračnu kde nejsou žádné objekty či nějaký shluk bodů. Tyto body nejsou definované jako velké plochy, proto je nemůžeme odstranit tak, jak odstraňujeme plochy. Často jsou jen malým shlukem bodů nebo jsou osamostatněny. Tyto body by pak následně mohli mít vliv na rychlost detekce, neboť se prezentují jako možné objekty po odstranění velkých ploch. Proto je potřeba tento problém řešit ještě než se odstraní velké plochy. Pak je mnohem jednoduší detekovat takovéto body pomocí prosté kontroly počtu bodů v sousedství a odstranění těch, které nedosahují určitých kritérií. Ty se určí podle vypočítání vzdáleností bodů od všech jeho sousedů. Vzdálenost bodu, která není v rozumném intervalu je možno prohlásit za outlier a odstranit z mračna. Ve výsledku to urychluje jak vyhledávání velkých ploch pro odstranění tak i práci klasifikátoru, které se nemusí zabývat těmito shluky, pokud nejsou odstraněny jiným způsobem.

Rozdělení mračna na klastry

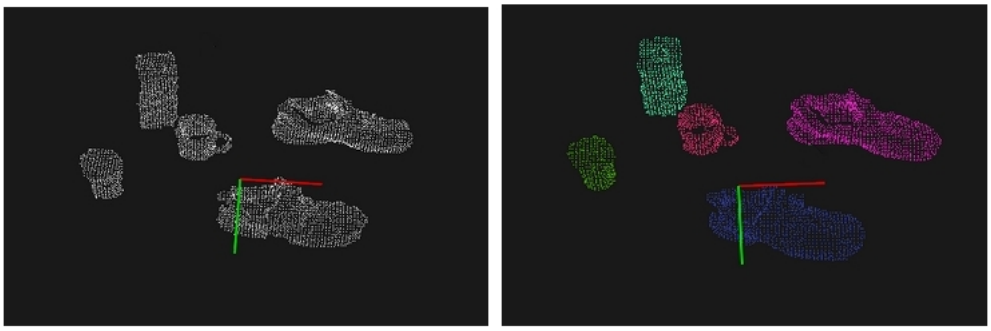
Po úspěšném odstranění ploch je v mračnu skupina potenciálním objektů. Tuto skupinu je potřeba rozdělit na jednotlivé objekty. Pro potřeby získání jednotlivých klastrů v mračnu jsem zvolil metodu Euclidean Cluster Extraction.

Princip metody Euclidean Cluster Extraction

Metoda je ve své podstatě velmi jednoduchá a její hlavní myšlenka je závislá na dané toleranci.

V podstatě si najdete bod v mračnu. Tento bod má nějakou svoji pozici v prostoru. Dále máme dvě pole, jedno je pro body, které budeme prozkoumávat někdy v budoucnu a druhé pole budou body, které jsme již prozkoumali. Tak tedy je základní bod umístěn do pole, pro body, které budeme teprve prohledávat a následně hledáme body v mračnu, které mají prostorovou vzdálenost mezi sebou a základním bodem menší než je zadaná vzdálenost. Když se nalezne takový bod, který má tu vzdálenost menší je umístěn do pole pro body, které budeme teprve prohledávat. Když už žádný bod nesplňuje podmínku, je základní bod umístěn do druhého pole a nebude se s ním dále pracovat. Následně se vybere z prvního pole bod, který je na nejnižším indexu. Prohlásíme ho za základní bod a znovu vyhledáváme body, pro které platí podmínka vzdálenosti, ale už nebereme v potaz body, které jsou buď v prvním nebo v druhém poli. Když konečně algoritmus dosáhne, že první pole je prázdné. To znamená, že se nalezly všechny body, které skládaly daný klastr. Následně pak při dalším

cyklu celého algoritmu hledáme bod, který není v druhém poli. Jak je takový bod nalezen, je následně přiřazen do prvního pole a druhé pole je vyprázdněno a klastr definován tímto polem je z mračna bodů odstraněn. Takto algoritmus opakujeme, abychom dostali všechny možné klastry v mračnu.



Obrázek 4.3: Celé mračno po odstranění ploch (vpravo) a po rozdělení na jednotlivé klastry v mračnu (vlevo).

4.3 Spočítání decriptoru objektu

Další části v návrhu detektoru je problém popisu geometrického tvaru objektů pro jeho klasifikaci. Když detektor získá klastry objektů, které chce rozpoznat, musí nejdříve nějak daný objekt popsat. Ve výsledném detektoru se pro tento popis používá VFH popsaný v kapitole 2.4.

Využití Point Feature Histogramu

První návrh detektoru využíval pro popis geometrických vlastnosti klastru metodu PFH. Tuto metodu z důvodu její velké složitosti 2.2 není možné využít.

Využití Fast Point Feature Histogramu

Tato část detektoru se často měnila a vyvíjela a další metodou, která byla v detektoru využita byla FPFH. V kapitole 2.3 je popsáno, že tato metoda je odvozeninou z metody PFH a snaží se dostat stejné popisovací síly jako jeho základní metoda. Především se ale snaží urychlit její výpočetní rychlost. I když se to FPFH metodě povedlo a dosáhla urychlení výpočtu a je ji možné využít při snaze dosáhnout detekce v reálném čase či téměř reálném čase. Problém této metody, ale byl především v souvislosti s využitým binárního klasifikátoru. Výsledkem FPFH je histogram hodnot. Bohužel tento histogram není vždy stejné velikosti, při popisu objektů tvoří pro každý bod 33 hodnot. Jelikož objekt získaný pomocí metody odstranění velkých ploch nemusí být vždy definován stejným počtem bodů. Využitý klasifikátor zde právě potřebuje již stejnou velikost histogramu pro své výpočty. Řešením tohoto problému je možné využití metody Global Fast Point Feature Histogram, která na základě vypočítaného FPFH a přiřazených kategoriích k bodům je schopná převést hodnoty histogramu na rovnocenné velikosti, ale není tak odolný vůči změnám měřítka daného objektu.

Využití Viewpoint Feature Histogramu

Výslednou metodou použitou v tomto detektoru je VFH [6], která umožňuje jak popis daného objektu na stejnou velikost, tak i výpočet má menší složitost a umožňuje tak popis objektu v reálném čase. Jak v případě FPFH je odvozeninou PFH, tak VFH se skládá z komponenty Histogramu vypočítaného z metody FPFH a komponenty hlediska. Liší se tedy hlavně komponentou hlediska, pod kterou se myslí senzor, kterým se scéna snímá. Především se využívá směr hlediska, tedy jeho vektory a normály, kterými je objekt popsán. Tím i tvoří vždy stejný počet hodnot pro popis daného objektu. Výhodu této metody je také její síla při změně měřítka objektu to znamená, že generuje stejná nebo podobná data pro stejný objekt při různých vzdálenostech od snímacího senzoru.

4.4 Klasifikátory

Poslední části detektoru jsou klasifikátory, které se snaží detekovat, o který objekt se jedná. Detektor jich využívá více, neboť klasifikátor, který jsem vybral je pouze binární, a to Support Vector Machines 2.5 [1]. Přesněji lineární klasifikátor, který na vstupu dostává výslednou hodnotu metody VFH, podle které klasifikuje objekty.

Jelikož je klasifikátor pouze binární to znamená, že určuje pouze, zda se jedná o daný objekt nebo ne. Jelikož je po detektoru požadována možnost rozpoznat více než jen jeden druh objektu je nutno vytvořit více než jeden klasifikátor.

Jeden proti všem

Z tohoto požadavku je tedy nutnost použít nějaký algoritmus, který rozhodne z hodnot všech klasifikátorů, o který objekt se jedná. A i když jsou metody jak použít jen jeden SVM klasifikátor pro více objektů ve výsledku se ukázalo jako nejjednodušší použít metodu souboje jednoho klasifikátoru proti všem ostatním. A vždy je prohlášen objekt podle hodnoty klasifikátoru, který je ten s největší hodnotou. I když je klasifikátor pouze binární je možné jej nastavit, aby vracel i přesné hodnoty svého výsledku a ne jenom zda hodnota překračuje daný práh a tím padem je prohlášena za kladný nebo negativní výsledek.

Kapitola 5

Implementace

Tato kapitola se zaměřuje na popis jednotlivých kroků detektorů, podle návrhu, z hlediska použitých algoritmů. Dále popisuje tvorbu datové sady pro klasifikátor a jeho strojové učení.

5.1 Využití Point Cloud Library

Point Cloud Library je knihovna pro 2D nebo 3D zpracování obrazu nebo mračna bodů[2]. V prvních několika krocích návrhu 4.1 se převážně využívá knihovny PCL. Tato knihovna umožňuje především nástroje pro práci s mračnem bodů a nabízí rozhraní pro komunikaci s Kinectem.

Zachycování dat z Kinectu

I když PCL umožňuje použití mnoha nástrojů, nemá algoritmy pro přímou komunikaci se senzory. Tento problém řeší pomocí rozhraní umožňující přístup k různým zařízením a jejich ovladačům, souborovým formátům a jiným zdrojům dat.

V této práci je využíván OpenNIGrabber, tedy rozhraní umožňující žádost o stream dat ze zařízení, která jsou kompatibilní s OpenNI [2]. OpenNI poskytuje svůj framework, který je kolekcí open source API (Application Programming Interface). Tyto rozhraní se snaží být standartami pro aplikace s přístupem k zařízením, která se snaží využít přirozené interakce (hlas, ruční gesta a sledování pohybu těla). Jelikož zde ale není potřeba se zabývat přirozenou interakci, neboť se snažíme zachycovat objekty. Využíváme OpenNI rozhraní pro zachycení vstupních dat detektoru. Kromě senzoru Kinect se dá využít i jiných senzoru jako je Asus Xtion Pro a Primesense Reference Design v rámci tohoto již implementovaného rozhraní.

V algoritmu detektoru je tedy potřeba vytvořit instanci OpenNIGrabber, která bude zajišťovat komunikaci s Kinectem. Když je zajištěna tato komunikace je potřeba zajistit získávání dat. Jako jednoduchý způsob bylo zvoleno vytvoření boost::bind objektu, kterému předáme adresu na callback funkce. Dále mu je přiřazena reference objektu, ve kterém je inicializován a placeholder (parametr pod číslem), kde by se měli nacházet data. Tento bind objekt je přiřazen k objektu boost::function, který je modelován na základě typu callback funkce. V této práci je tento datový typ z knihovny PCL [2], který reprezentuje datový typ mračna bodů. Jedna se o jednoduchý C++ vektor, který obsahuje datatyp jednotlivých bodu. Těchto datových typů je v PCL mnoho a převážně se liší vlastnostmi, které mohou definovat bod v prostoru. Například jenom souřadnice x, y, z či je ještě možné použít r, g, b hodnoty a další. OpenNIGrabber není pouze limitován na datový typ mračna bodů, ale

je schopný získat hloubkovou mapu nebo i prostý 2D obrázek z Kinectu. Výsledný objekt funkce je pak možné registrovat pomocí instance OpenNIGrabber a následně nastartovat. Ve výsledku tak dostáváme v nabindované callback funkci parametr datového typu, který je určen v boost::function, jenž byl získán z dat Kinectu.

Třída deskriptor

Tato třída obsahuje funkční část projektu pro práci v mračnu bodů. V podstatě se jedná o implemetaci objektu, která provádí první 3 části návrhu. Zapouzdřuje funkce `planarExtraction` pro extrakci ploch, `clusterExtraction` pro získání jednotlivých klastru ke klasifikaci. Dále obsahuje funkci `vfh` pro vypočet hodnoty VFH pro určitý klastr. Jednotlivé principy metod jsou vysvětleny níže.

Tato třída spolupracuje s třídou `classifier`, která se stará o klasifikaci. Ta vrací informaci o jaký objekt se jedná pro možné zvýraznění v mračnu.

Extrakce velkých ploch

V sekci 4.2 zabývající se návrhem detektoru je popsáno proč se odstraňují jednotlivé plochy. Zde je naopak popsáno jak je možné této extrakce dosáhnout v algoritmu detektoru s využitím PCL [2].

Když tedy máme data z Kinectu v určeném datovém typu reprezentujícím mračno bodů. Nemůžeme s ním pracovat hned, neboť je potřeba mít na vědomí, že při využití zmíněného postupu získáváme konstantní ukazatel na hodnotu snímaného mračna. Proto je potřeba nejdříve hodnoty daného mračna transformovat do datového typu, který nám umožňuje jeho manipulaci.

Odstranění ploch v PCL

Prvním krokem za cílem odstranění velkých ploch je vytvoření modelu plochy. Je zde možno definovat jakýkoliv tvar, který bychom chtěli odstranit z mračna. Zde se ale zaměříme na plochu. Autoři PCL již nabízí modely základních geometrických tvarů a přímo i třídu, která je určená pro segmentaci a je jí možno přiřadit jak model, tak i metodu. Vytvoří se tedy instance tohoto objektu a je jí přiřazen model plochy. Dále se přiřadí i hodnota prahu pro vzdálenost, tato hodnota rozhoduje, zda bod ve vzdálenosti je inlier, tedy je jeho pozice vysvětlená pomocí vlastností modelu plochy. Poslední vlastností objektu pro segmentaci je metoda, která definuje plochu v mračnu. V projektu jsem zvolil iterativní metodu RANSAC.

Když máme objekt, který nám reprezentuje plochu je možno provést i extrakci takto definovaných objektů v mračnu bodů. PCL tu nabízí další objekty a způsoby možné extrakce. Já jsem zde použil objekt pro extrakci indexů z určeného datového typu v tomto případě prostorem mračnu bodů s vlastnostmi pouze souřadnic. Objektu je předáno vstupní mračno, ze kterého je potřeba odstranit dané modely. Dále se objektu musí určit i množina indexů, kam se ukládají indexy bodů, které nebudou extrahovány. Dále se musí nastavit, zde chceme extrahovat modely, nebo chceme odstranit jejich okolí. V poslední řadě je potřeba ještě objektu dát ukazatel na výslednou proměnou, která by měla být stejného datového typu jako je vstupní mračno.

Rozdělení klastrů

Pokud je extrakce ploch úspěšná dostaneme mračno bodů, které by mělo vypadat, jak ukazuje obrázek 4.2. Momentální mračno by se mělo tedy skládat z možných objektů, které

jsou ale součástí jednoho mračna. Je tedy potřeba tyto objekty od sebe oddělit. Pak je možné pro každý objekt spočítat jejich deskriptor a následně je nechat prověřit klasifikátory.

Euclidean Cluster Extraction v PCL

V PCL už je na tuto metodu připravených několik objektů, které maskují princip této metody. Samozřejmě se nejedná o přesný princip, jak je popsáno v sekci výše, ale o přizpůsobenou metodu pro účely PCL [2].

Tak tedy v PCL je potřeba pro urychlené vyhledávání využít nějakého objektu, který dokáže dané mračno popsat a umožnit tak rychlé vyhledávání. Pro tyto účely se často využívá datová struktura pro popis prostoru na části a organizování bodů v k-dimenzionálním prostoru zvaná K-d tree. Jedná se o speciální verzi metody binárního rozdělování prostoru. Tomuto objektu pak dáme mračno, které není rozdělené na jednotlivé klastry [10].

Dalším objektem, který je v PCL požadován pro extrakci klastrů je vektor pro datový typ indexů. Tento datový typ zase není nic jiného než další vektor datového typu integer. Do tohoto vektoru jsou ukládány všechny indexy všech klastrů. To znamená, že na indexu 0 je vektor, který obsahuje indexy všech bodů pro daný klastr.

Když je inicializován objekt pro prohledávání mračna a pro ukládání je možno inicializovat objekt pro Euclidovu extrakci. Tomuto objektu musíme hlavně přiřadit velikost. Jakou mají mít body od sebe maximální vzdálenost, aby se dalo prohlásit, že jsou oba ve stejném klastru. Dále se mu musí, přiřadí metoda pro prohledávání mračna, tedy objekt K-d tree. Samozřejmě je ještě potřeba mu určit vstupní mračno. Jsou zde také další možné vlastnosti, jako jsou minimální a maximální velikosti klastrů. Obzvlášť minimální velikost by se měla zadávat, aby se nebrali objekty s příliš malým počtem bodu, které můžou být jen pozůstatkem po metodě odstraňování ploch. Následně je už při extrakci předán objektu vektor, kam se uloží indexy jednotlivých klastrů.

Na konec je potřeba rekonstrukce jednotlivých klastrů z vektoru obsahujícího indexy zpět na datový typ mračna bodů. Výsledné mračno je pak předáno funkci pro výpočet VFH hodnoty.

Viewpoint Feature Histogram

Ze sekce 2.4 víme princip a jednotlivé komponenty VFH. Teď se ale podíváme jak naimplementovat tento proces popisu objektu v rámci knihovny PCL.

VFH implementace využívá 45 intervalů histogramu pro každou hodnotu FPFH plus dalších 45 pro vzdálenosti mezi každým bodem a hlediskem. A dalších 128 je určených pro komponentu směru senzoru. To má za následek 308 bytové pole datového typu float. To je v PCL reprezentováno datovým typem `VFHSignature308`. Berme ale na vědomí, že se jedná pouze o jedno pole o velikosti 308, kdežto při použití FPFH se počítá deskriptor pro každý bod.

Implementace VFH v PCL

Implementace VFH je velice jednoduchá, neboť jsou tu již objekty, které se o většinu algoritmu postarají samy. Je jen potřeba je správně nastavit.

Jedním z těchto objektů je objekt reprezentující normály. Ze sekce 2.2 jde zjistit, jak se vypočítávají. V PCL jsou zde již algoritmy implementované a znovu stačí nastavit správné objekty na správné hodnoty. Je tedy potřeba inicializovat objekt pro odhad normál `NormalEstimation`, který potřebuje vědět vstupní a výstupní datovou strukturu. Tomuto

objektu se tak musí přiřadit vstupní mračno bodů. Jelikož je výpočet normál závislí na vyhledávání bodů ve svém okolí, algoritmus tak využívá vlastností datové struktury Kd-tree pro rychlé vyhledávání. Dále se zde musí také nastavit velikost sousedství, což je důležitá hodnota z důvodu rozlišení objektu (více v sekci 2.2).

Když je algoritmus pro výpočet normál správně implementován, je možno využít těchto povrchových normál pro tvorbu deskriptoru. Je zde znovu potřeba inicializovat jednotlivé objekty, do kterých buď ukládáme jednotlivé hodnoty deskriptorů, nebo nám provádí výpočet VFH. Je tedy potřeba inicializovat datovou strukturu `VFHSignature308` pro uložení výsledného deskriptoru a objekt `VFHEstimation`, pro výpočet deskriptoru. Tento objekt potřebuje ke svému výpočtu, kromě vstupního mračna a normál pro popis povrchu objektu, objekt pro vyhledávání v mračnu. Jedná se tedy o další instanci třídy Kd-tree.

5.2 Klasifikace pomocí OpenCV

OpenCV, nebo-li Open source Computer Vision, je knihovna která se zabývá, jak už z názvu vypovídá, počítačovým viděním. Je možno s ní pracovat jak v C, C++, Pythonu tak i v Javě [5]. Je také podporovatelná na Windowsu, Linuxu, Mac OS, iOS a Androidu.

Knihovna OpenCV je v tomto projektu využívaná výhradně pro klasifikaci objektů a skladování dat do jednotlivých souborů. V kapitole 2.5 je popsán princip binárního klasifikátoru Support Vector Machines, který je v této práci využíván. Tento klasifikátor je již v knihovně OpenCV vytvořen včetně algoritmů pro jeho strojové učení a predikování výsledku.

V algoritmu detektoru je využito několik klasifikátorů, které bojují mezi sebou, aby mohli prohlásit, že objekt popsán VFH metodou patří k nim.

Třída classifier

V projektu je klasifikace (objekt SVM) součástí třídy `classifier`, která spolupracuje s třídou `descriptor`. Od této třídy získává deskriptor objektu pro klasifikaci. Funkce starající se o určení klasifikace objektu je funkce `predict`, která vrací třídě `descriptor` informaci jak daný klastr z mračna klasifikovala.

Strojové učení

Pro každý objekt bylo nutné vytvořit jeho příslušný SVM klasifikátor. Tento klasifikátor bylo nutné naučit rozpoznávat jeho příslušný objekt od ostatních. To se dělalo pomocí již předem vypočítaných deskriptorů objektů. Ke každému tomuto deskriptoru bylo určeno, zda se jedná o daný objekt nebo ne.

Aby nebylo nutné každý tento klasifikátor učit ručně, byl vytvořen skript, který hodnoty do jednotlivých deskriptorů převedl do jedné matice a přitom vytvářel další matici, která obsahovala hodnoty 1 nebo 0 určující zda se jedna o daný objekt nebo nikoliv. Tyto matice pak byly předány algoritmu, který učil příslušný klasifikátor.

Kapitola 6

Experimenty klasifikátorů

Experimenty popisované v této kapitole jsou prováděny nad jednotlivými objekty z testovací sady a klasifikátory, které byly naučeny pomocí vytvořené datové sady.

6.1 Datová sada

Jelikož je detektor zaměřený na objekty, které se nacházejí na stole, je i datová sada založená ne této myšlence. Vybral jsem tedy 4 základní objekty, které se na stůl vlezou a jsou reprezentací různých tvarů.

Tyto objekty jsou hrnek, krabice, láhev a talíř. Aby byla možná detekce těchto objektů pomocí klasifikátoru SVM, je potřeba každý objekt několikrát nafotit a pro každý vypočítat jeho VFH hodnoty. Ty ho následně naučit.

Vytvořil jsem tedy přibližně 170 až 220 fotek každého objektu (nejméně 2 různé druhy toho objektu), které se převedli do datové struktury mračna bodů. Těchto 700 fotek bylo foceno, tak abych nemusel každou ručně projíždět a kontrolovat, zda se jedná o daný klastr, který chci (tedy daný objekt). Bylo zde tedy využito již naimplementovaných vlastností detektoru a to odstranění jednotlivých velkých ploch. Scéna se tedy skládala ze dvou ploch proti, kterým byl objekt focen. Přesto bylo potřeba objekt nafotit v tomto prostředí pod různými úhly a v různých rotacích objektu.

Pro vyhodnocení deskriptoru byl vytvořen script, který spouštěl část detektoru, která prováděla výpočet tohoto deskriptoru a pouze zachycoval nejbližší klastr s počtem bodů větší než 1000. Díky tomuto bylo možné celou datovou sadu vytvořit automatizovaně. Výsledné deskriptory byly tak ukládány pomocí Knihovny OpenCV, která má implementovány objekt, který má rychlé funkce pro zápis a čtení YML souborů.

Pomocí tohoto scriptu bylo zautomatizováno vytváření deskriptoru a ukládání. Přesto se nejednalo o 100% funkčnost, ale bylo potřeba výsledek zkontrolovat. Proto jsem ukládal společně s deskriptorem i mračno bodů, které tento deskriptor popisoval. Při zobrazení tohoto mračna tedy bylo hned zjevné, zda script uložil správný deskriptor či nikoliv.

6.2 Testovací sada

Výsledná testovací sada se také jako datová sada pro klasifikátory, skládá ze 4 objektů (hrnek, láhev, talíř a krabice). Předtím než se jednotlivé klasifikátory začali učit, tak jsem náhodně z každé sady 180 vypočítaných deskriptorů vyřadil 10 až 15, které slouží jako

testovací sada. Následně jsem nafotil dalších objekty, aby bylo aspoň 25 testovacích objektů na objekt.

Tedy testovací sada se skládá přes 100 deskriptorů VFH pro 4 objekty (2 druhy ke každému), kde každý objekt má 25 kusů různých deskriptorů pro různé rotace objektu pod různými úhly.

6.3 Návrh experimentů

Jednotlivé experimenty budou provedeny nad 4 klasifikátory. Každý klasifikátor je zaměřen na jeden objekt z datové sady, jmenovitě se jedná o objekty hrnku, láhve, talíře a krabice.

Experimenty by měli ukázat vlastnosti jednotlivých klasifikátorů a také sílu vůči ostatním klasifikátorům. Experimenty se budou provádět nad celou testovací sadou.

6.4 Experiment jednotlivých klasifikátorů

V tomto experimentu jsou jednotlivé klasifikátory postaveny proti testovací sadě. Jejich výstup byl, zaznamenám v ROC křivce. V ní by mělo být vidět, který klasifikátor je nejpřesnější a který naopak je nejméně přesný.

U experimentu jsem předpokládal, že ukáže klasifikátor pro objekt krabice jako nejpřesnější. Jedná se totiž o jediný objekt, který nemá zaoblený tvar. Ostatní klasifikační objekty mají geometrický tvar, který je zaoblený a proto je předpokládano vítězství klasifikátoru krabice. Druhý nejpřesnějším klasifikátorem je předpokládán talíř, jelikož geometrické tvary hrnku a láhve jsou si podobné. U hrnku a láhve je předpokládána nejslabší míra doopravdy pozitivních. Důvod k tomuto předpokladu je, že jejich základní geometrický tvar si je velice podobný a oba tvoří válec.

Z důvodu nemožnosti nastavit rozdělovací práh ve využitém klasifikátoru z knihovny OpenCV jsem využil funkce predict, která může vracet hodnotu vzdálenosti od optimální nadroviny klasifikátoru. Tato hodnota určuje jak moc si je daný klasifikátor jistý, že se o daný objekt jedná nebo nejedná.

Proto bylo prováděno imaginární posouvání optimální nadroviny pro výpočet hodnot True Positive (TP), False Negative (FN), False Pozitive (FP) a True Negative (TN) jak je popsáno v 2.5. Tento výpočet se musel, prováděl pro každý posun nadroviny. Tento přepočet se musel provádět, jelikož se klasifikátor důsledkem posunu nadroviny stával buď více přísným ve svém rozhodování (udělal méně chyb) nebo naopak je mu jedno o jaký objekt se jedná (udělá více chyb).

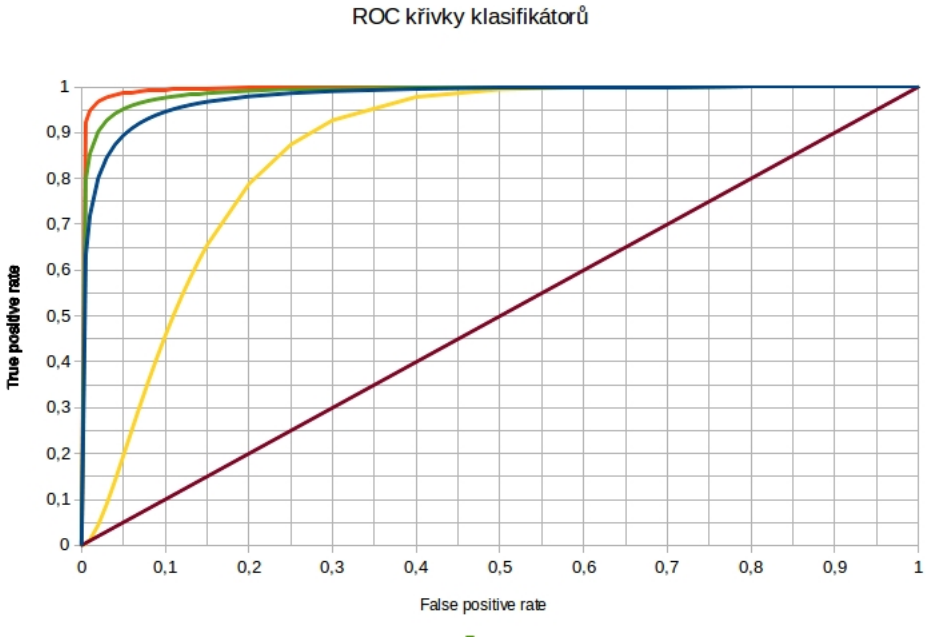
Z těchto hodnot se pak vypočítali hodnoty senzitivity a specificity 2.5. Pomocí hodnot senzitivity a specificity v každém posunu rozhodovací nadroviny byly určeny jednotlivé hodnoty bodů tvořící příslušnou ROC křivku 6.1.

Vyhodnocení experimentu

V grafu 6.1 jde vidět, že klasifikátory hrnku, talíře a láhve dopadl experiment dobře. Tyto klasifikátory ukazují, že již při nízké hodnotě rozhodovacího prahu mají velkou míru správně pozitivních okolo 90%. Z hlediska předpokladu se ukázal klasifikátor talíře doopravdy přesnější než klasifikátory hrnku a láhve.

Naopak jako nejméně přesný se ukázal klasifikátor krabice, který nedosahuje kvalit ostatních klasifikátorů. I když nedosahuje takových kvalit, může se konstatovat, že pracuje správně. Jeho reálný predikátor, jak je v grafu vidět, je nad náhodným klasifikátorem. Ten

je zobrazen pro ukázání, že klasifikátor krabice má větší přesnost (křivka je zobrazena nad náhodným klasifikátorem) než je náhodný.



Obrázek 6.1: ROC křivky klasifikátorů objektů talíř (červená), láhev (zelená), krabice (žlutá), hrnek (modrá) a náhodného klasifikátoru (hnědá).

6.5 Experiment detektoru s klasifikátory

V tomto experimentu se budou testovat všechny klasifikátory najednou. Budou mezi sebou bojovat o objekty z testovací sady. Hodnoty klasifikátoru jsou získány z objektu SVM z knihovny OpenCV.

Klasifikátory budou mezi sebou bojovat ve stylu jeden proti všem. Touto metodu je určen vítězem klasifikátor, který si je nejvíce jistý svojí predikcí. Tím porazí protivníky. Jednotlivé predikce se hodnotí vůči optimální nadrovině, jak ji určil algoritmus v objektu SVM z OpenCV.

V experimentu budou detektoru předávány různé mračna skládající se pouze z předem připravených objektů. Objekty, které budou předávány, jsou z testovací sady. Pokud detektor správně pomocí klasifikátoru rozpozná jeden ze 4 klasifikačních objektů, označí jej buď červenou pro láhev, žlutou pro hrnek, zelenou pro talíř, modrou pro krabici a bílou pro nerozpoznání objektu.

Z výsledků minulého testu by měl být nejslabší klasifikátor krabice. Jelikož byl vyhodnocen jako nejméně přesný klasifikátor, tedy měl velkou míru špatně pozitivních jedinců. To znamená, že často si byl jistý svoji volbou, i když se nejednalo o jeho objekt. Pokud se bude vycházet z této myšlenky i u ostatních klasifikátorů, mělo by být pořadí 1. klasifikátor talíře, 2. klasifikátor pro láhev a předposlední pro hrnek.

Experimenty

Ukázané experimenty jsou jen částí z celé testovací sady. Experiment ukazuje specifické vlastnosti jednotlivých klasifikátorů v detektoru.

Obrázek 6.2: Neklasifikovaný objekt (vlevo) a klasifikovaný (vpravo), který je obarvený na žluto (klasifikován jako hrnek).

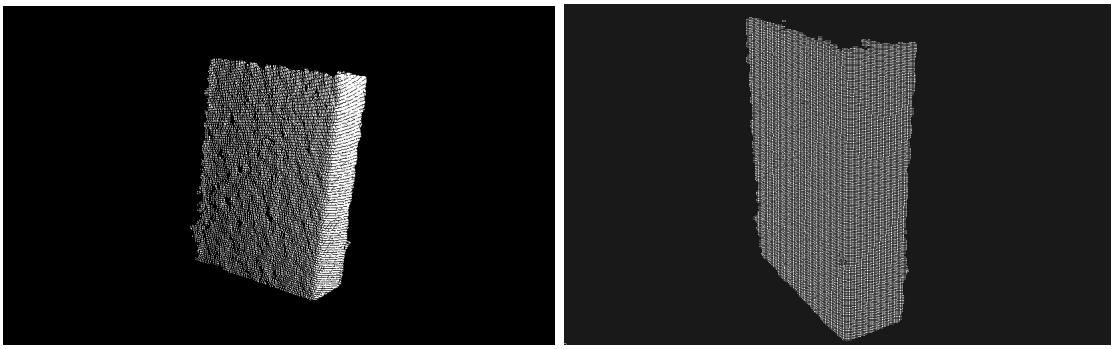
V obrázku 6.2 je první klasifikační objekt hrnek. Pro vstupní mračno vlevo byl počítán deskriptor. Následně byl pak pomocí klasifikátoru na základě deskriptoru klasifikován jako hrnek (detektor ho obarvil na žluto).

Všimněte si, geometrických vlastnosti hranku. První věcí je, že jeho tělo není čistý válec. Hrnek má podstavu užší než je jeho hrdlo. Další vlastností je jeho ucho, které má na vrchu zaoblený tvar ale dole pouze splývá s hrnek.

Obrázek 6.3: Neklasifikovaný objekt (vlevo) a klasifikovaný (vpravo), který je obarvený na žluto (klasifikován jako hrnek).

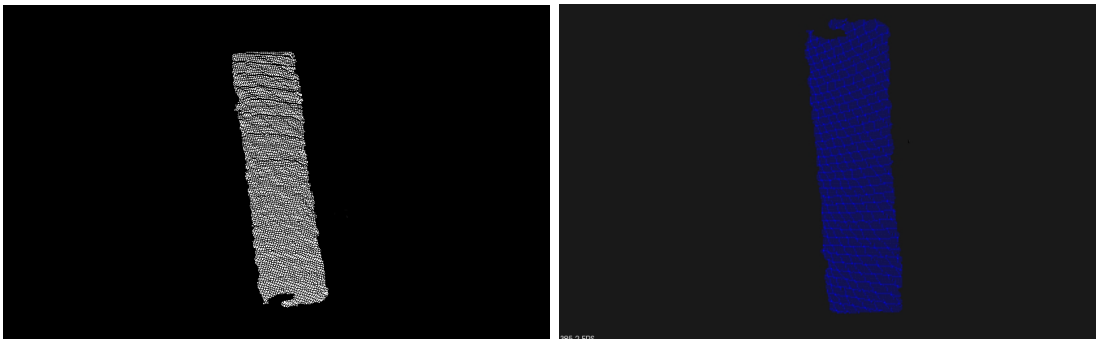
V tomto experimentu byl detektoru poslán znovu hrnek. Geometrický tvar tohoto hrnku, ale nebyl shodný s hrnekem předešlým. Tento hrnek má jiný tvar těla a jeho podstava je tím pádem stejně široká jako jeho hradlo hrdlo. Ucho má dole více zakulacený tvar vůči předchozímu hrnku.

Klasifikátor pro objekt hrnku se v experimentu ukázal být schopen správné detekce pro své vlastní objekty. A to i když byli v testovací sadě dva různé druhy stejného objektu (hrnku). Byl jsi schopný obhájit svůj objekt a také neměl potřebu klasifikovat jiné než svoje objekty.



Obrázek 6.4: Neklasifikovaný objekt (vlevo) a klasifikovaný (vpravo), který je obarvený na bílo (klasifikován jako neznámý).

Druhým klasifikátorem je klasifikátor krabice. Z geometrického hlediska je krabice velmi jednoduchým objektem z klasifikovaných objektů. Jedná se totiž jenom o plochy, které jsem na sebe kolmé. Je tedy s podivém, že klasifikátor tohoto objektu nebyl v některých případech schopen klasifikovat vlastní objekty, i když se k nim nikdo jiný nehlásil.

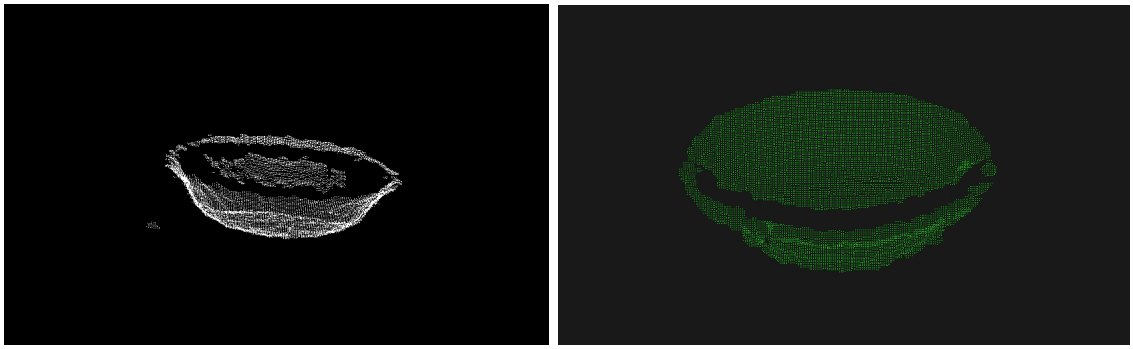


Obrázek 6.5: Neklasifikovaný objekt (vlevo) a klasifikovaný (vpravo), který je obarvený na modro (klasifikován jako krabice).

Jelikož klasifikátor pro objekt krabice nebyl moc úspěšný, podíval jsem se na případ doopravdy pozitivního klasifikování. Pomocí toho jsem zjistil, že klasifikátor nemá problém klasifikovat objekty při zobrazení jenom jedné plochy. A také bylo zvláštní, že tento klasifikátor byl velice náchylný k přebýrání jiných objektů. Důvod je pravděpodobně v datové sadě pro tento objekt, která byla na základě tohoto experimentu upravena.

Dalším klasifikačním objektem je talíř 6.6. Talíř stejně jako hrnek je schopen obhájit svoje objekty poměrně dobře s velkou procentuálí úspěšnosti. A nezajímá ho žádný jiný než jeho vlastní třída objektů. Jediný problém, který měl talíř, byl s klasifikátorem krabice.

Posledním objektem ke klasifikaci byla láhev. Ta také jak hrnek i talíř neměla problém udržet si svoje objekty vůči ostatním. Pravděpodobně pro svůj specifický geometrický tvar, kde buď je v datové sadě definovaná jako kompletní láhev s válcovitým podstavcem a zúžením na vrchu. Nebo na dva nespojené klastry. Ty vznikají, když snímaná láhev je skleněná nebo plastová a má etiketu a vršek nebo korek. Kinect není totiž povrch plastu nebo skla není schopen zachytit.



Obrázek 6.6: Neklasifikovaný objekt (vlevo) a klasifikovaný (vpravo), který je obarvený na zeleno (klasifikován jako talíř).

Vyhodnocení

Testovací sada byla pro tento experiment o něco rozšířená, aby se dosáhlo o trochu přesnějších hodnot.

Objekt	Počet prvků	Správně klasifikovaných	Přesnost
Hrnek	40	35	87,5%
Láhev	40	31	77,5%
Talíř	40	40	100%
Krabice	40	10	25%
Celkově	160	116	72,5%

Tabulka 6.1: Tabulka úspěšnosti klasifikátoru objektů predikovat vlastní objekty.

Tabulka 6.1 ukazuje úspěšnost jednotlivých klasifikátorů při predikci vlastních objektů. Je také ukázana celková pravděpodobnost správné klasifikace v celkovém detektoru.

Tabulka 6.2 zase ukazuje negativní hodnoty systému klasifikátorů. Snažíme se zde dosáhnout co nejmeších hodnot, aby klasifikátor byl co nejpřesnější.

Objekt	Negativní jedinci pozitivně	Pozitivní jedinci negativně	Neklasifikováno
Hrnek	2	5	0
Láhev	11	9	0
Talíř	0	0	0
Krabice	13	12	15
Celkově	26	26	15

Tabulka 6.2: Výskyt špatných klasifikací objektů.

Když se zaměříme na jednotlivé procentuální úspěšnosti predikce klasifikátorů svých objektů, jednoznačně vede klasifikátor pro objekt talíř. Tento klasifikátor dokázal klasifikovat svoje objekty se 100% úspěšností. Druhým nejlepší je klasifikátor hrnku s 87,5% procentní úspěšností. Předposlední je klasifikátor pro láhev s 77,5% úspěšností. Poslední je klasifikátor krabice s pouhými 25%. Tento klasifikátor i přes upravení datové sady není schopen dosáhnout lepších výsledků. A pravděpodobně je potřeba předělat celou tuto sadu. Celková úspěšnost klasifikátoru je tak odhadována na 72,5%.

Kapitola 7

Závěr

Tato bakalářská práce popisuje metody pro popis objektů zachycených senzorem Kinect a ukazuje jak využít tyto metody pro klasifikaci těchto objektů. Tyto znalosti byly využity k vytvoření aplikace, která pomocí senzoru Kinect zachytává snímané okolí, v kterém vyhledává možné objekty. Pro nalezené objekty se snaží vytvořit deskriptor metodou Viewpoint Feature Histogram. Jenž je využit pro klasifikaci objektů pomocí binárního klasifikátoru Support Vector Machine. Cíle stanovené v zadaní v úvodu této technické zprávy a v pokynech této práce byly dosaženy a detektor byl implementován. Provedený experiment nad výsledným detektorem podává úspěšnost detekce doopravdy pozitivních objektů nad testovací sadou okolo 72,5%.

Aplikace ovšem není dokonalá a je ji možné ještě dále rozšiřovat či vylepšovat. Detektor by mohl dostat více naučených objektů a rozšířit tím jeho datovou sadu. Rozšířením datové sady detektoru by pak měl více oblastí využití. Obzvlášť by potřeboval vylepšit klasifikátor pro objekt krabice, který je podprůměrem úspěšnosti detekování doopravdy pozitivních objektů. Tím způsobuje i pokles kvality celého detektoru.

Vylepšit by se také mohla část detektoru zabývající se se segmentací. Tato část pracuje sice správně, ale provádí se nejdelší dobu při detekci. Možné řešení by bylo využití filtrů, které by pak ale měli negativní dopad na deskriptor objektu.

Celkově tato práce může sloužit jako základ pro systém robota, který by měl k sobě připojen senzor Kinect. A pomocí vlastnosti zpracování v mračnu bodů tak získávat možné geometrické vlastnosti objektu, jako je jeho průměr nebo jeho vzdálenost od senzoru.

Literatura

[1] Gary Bradski, A. K.: *Learning OpenCV*. O'Reilly Media, Inc, 2008, [cit. 2014-05-13], iSBN 978-0-596-51613-0.

[2] Library, P. C.: PCL - Point Cloud Library [online]. [online], http://www.pointclouds.org, 2014 [cit. 2014-05-13].

[3] Marco Zuliani: RANSAC for Dummies. [online], Dostupné na http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf, 2012 [cit. 2014-05-11].

[4] Microsoft: Kinect for Windows Sensor Components and Specifications, [cit. 2014-05-12]. [online], http://msdn.microsoft.com/en-us/library/jj131033.aspx.

[5] OpenCV: OpenCV – OpenCV. [online], http://www.opencv.org, 2014 [cit. 2014-05-11].

[6] R.B. Rusu, G. Bradski, R. Thibaux, J. Hsu: Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram. In *International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010, [cit. 2014-05-9].

[7] R.B. Rusu, N. Blodow, M. Beetz: Fast Point Feature Histograms (FPFH) for 3D Registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 1999, [cit. 2014-05-08].

[8] R.B. Rusu, N. Blodow, Z.C. Marton,m. Beetz: Aligning Point Cloud View using Persistent Feature Histograms. In *21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008, [cit. 2014-05-07].

[9] Rusu, R. B.: *Semantic 3D Object Maps for Everyday Manipulation in Human Living Enviroments*. Dizertační práce, der Technischen Universität München eingereicht und durch die Fakultät für Informatik, 2009, [cit. 2014-05-07].

[10] Wikipedia: K-d tree – Wikipedia, the free encyclopedia, [cit. 2014-05-6]. [online], http://en.wikipedia.org/wiki/K-d_tree.

[11] Wikipedia: Receiver Operating Characteristic – Wikipedia, the free encyclopedia, [cit. 2014-05-13]. [online], http://en.wikipedia.org/wiki/Roc_curve.

Příloha A

Obsah CD

- Zdrojové kódy aplikace
- Datová a testovací sada
- Elektronická verze této zprávy
- Plakát

Příloha B

Manual

Pro funkčnost aplikace je nutné mít knihovnu OpenCV ve verzi 2.4.5 nebo novější a také knihovnu PCL ve verzi 1.7. Instalace knihoven je popsána na jejich příslušných webových stránkách (`http://www.pointcloud.org` pro PCL a `http://www.opencv.org` pro OpenCV).

Pro úspěšné zkompilování všech zdrojových souborů je potřeba mít nainstalovaný cmake. Pro zkompilování programu stačí spustit script `compile.sh`, který provede příkazy cmake a make. Následné spuštění se provádí souborem `projekt`, který je uložen ve složce `bin`. Soubor přímá parametr `-f`, který určuje zda aplikace bude brát data o mračnu bodů ze souboru pcd nebo přímo z kinectu. Dále jsou možné paramtry `-m` pro určení minimalní velikosti klastru, za tímto paramtrem musí naslednovat číselná hodnota. Pak je parametr `-p` pro vypnutí extrakce ploch (pokud už je mračno filtrované).

Parametry:

- -f: pro určení vstupního souboru,
- -p: vypnutí extrakce ploch,
- -m: pro určení minimální velikosti klastru.

Příklad spuštění:

- pro práci s kinectem: ./projekt
- pro práci se souborem: ./projekt -f soubor.pcd
- příklad všech parametrů: ./projekt -f soubor.pcd -m 300 -p

Příloha C

Plakat